Final Report for

Representations and Protocols for Universal Access to the World-Wide-Web

January 31, 2001

Sponsored by

Defense Advanced Research Projects Agency (DOD)

Arlington, VA

DARPA Order D611/76

Issued by U.S. Army Aviation and Missile Command Under

Contract No. **DAAH01-00-C-R128**

Phase I SBIR Topic Number: SB001-020

Progeny Systems Corporation
8809 Sudley Road, Suite 101
Manassas, VA 20110-4716
(703) 368-6107

Contractor: Gary J. Sikora Principal Investigator

Data Item: 0001AB
Reporting Period: 04 April 2000 - 4 December 2000
Effective Date of Contract: 04 April 2000
Contract Expiration Date: 04 December 2000

UNCLASSIFIED

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED

Table of Contents

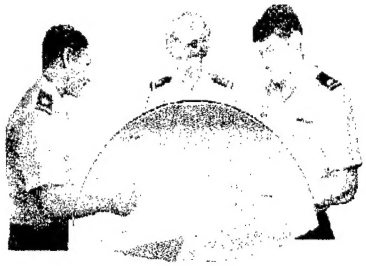
List of Figures

Representations and Protocols for Universal Access to the World-Wide-Web

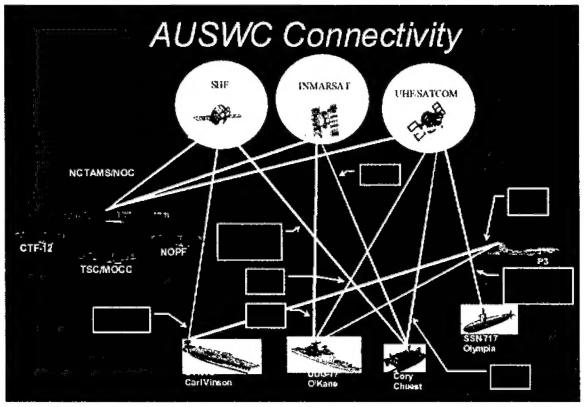
January 31, 2001

1 Abstract

Information integration and aggregation from a variety of sources, including Command and Control (C^2) systems, reconnaissance data, satellite data, unit capability data, logistical data and real-time battlefield conditions, with distribution to users at all echelons has been pursued for some time. Developments have been slow for several reasons including funding priorities, previous false starts, and the gap between commercial technologies and the Defense specific needs. Several factors may now allow information-centric warfare to become reality. First, the developments under DARPA, Information Technology Office, Intelligent Software, Communicator research program has lead the commercial industry in areas of speech recognition and interaction providing real-world technology evaluation. Second, the US Air Force Scientific Advisory Board (SAB) with the Joint Battlespace Infosphere (JBI) program reinitiated a battlefield common picture in 1998. Unlike previous science board studies, SAB provides enough technical detail and direction for the services to begin JBI development. Thirdly, Web technology has advanced so fast over the last five years that information solutions have become easier and much more affordable. To this end, the effort under this Phase I project was to research, prototype and validate Web technologies to obtain a multi-modal, common-authoring, universal access to the World-Wide-Web for the wireless industry and hands-free applications, that supports the DARPA Communicator and US Air Force SAB JBI missions.



The opportunity is to close the gap between commercial Web technology and battlefield information access infrastructure. With a forward-looking approach, the technology gap is reduced to a point where a buy-integrate-test development cycle is realizable. Recent XML and speech technologies and third generation device developments are the key enablers with the use of agents and connectors to mitigate COTS product alterations. Progeny Systems Corporation is contracted to do similar type work for the Navy Advance Undersea Warfare Concepts (AUSWC) network-centric applications. We are interconnecting sensor systems across the Navy's war fighting assets using Web technologies. This work is being performed under a Navy Phase III SBIR contract with congressional plus-up money.



Soon the technology gap between commercial Web and the battlefield information-centric requirement will be closed—except for special interfaces requiring connectors, gateways, etc. Today the Web is a paging experience with links to other pages. Push technologies providing such things as news services have been around for a couple years. Now with broadband and XML the commercial trend will migrate the Web experience to an interactive, information-based push technology where the user can build information, request notification and collaborate. This trend has already started in cell phone application. The transportation, industrial and investment communities can benefit by this technology, hence will drive these standards. This provides a great opportunity for both the DARPA and SAB initiatives, as well as further improving commercialization potential.

2 Introduction

The research was focused on elements of process and product to optimize Return-On-Investment (ROI), cost-of-ownership and interoperability for the DARPA Communicator and JBI developments, and other commercial initiatives. The focus on process was to ensure resulting capability implementations are of current, commercial open-standards that allow low-cost technology refresh. A complete, functionally designed system that comprises stove-piped solutions, age-locked technologies, or ones that are not on the commercial mainstream falls far short of a solution that will last generations—information interoperability will reappear at the next technology turn. Instead, the research aligned with commercial trends, and used technologies with no modifications, and involved collaboration with the industry leaders in search of emerging technologies.

VoiceXML FORUM

Early on in the research, the VoiceXML standard was created by a joint effort of Lucent, Motorola, AT&T and IBM from their individual speech markup technologies. Because of this alliance and that these four companies are considered the industry leaders in speech technologies and integrated server solutions, we considered VoiceXML a technology that will last. Given this, VoiceXML became an integral part of the solution and we decided to collaborate with all four companies to search out relationships that together can help develop the remaining components in providing universal access to the World Wide Web. IBM proved to have the technology and vision that aligned with ours to make this happen.

W3C XSL Transformations (XSLT)

Our research and prototyping was neck-to-neck with commercial industry's trend of using XSLT and other proprietary techniques to publish Web content on emerging devices as illustrated in Figure 1. Often our research pointed to solutions that where subsequently published by O'Reilly online publications or vendor products such as Oracle's Portal-to-Go. Consequently, many solutions are converging upon multi-channel techniques such as Web sites proxied by PDA portals. In attempts to gain early market share, Internet access, multi-modal, cost-of-ownership and interoperability have been compromised.

The proposed Phase II research and development addresses the future Web model that supports multi-modal and common source, as illustrated in Figure 1. The target platforms include both desktop and 3rd generation devices that can host the required processing for multi-modality interaction. At this juncture commercial technologies will have converged with the needs of battlefield information accessibility.

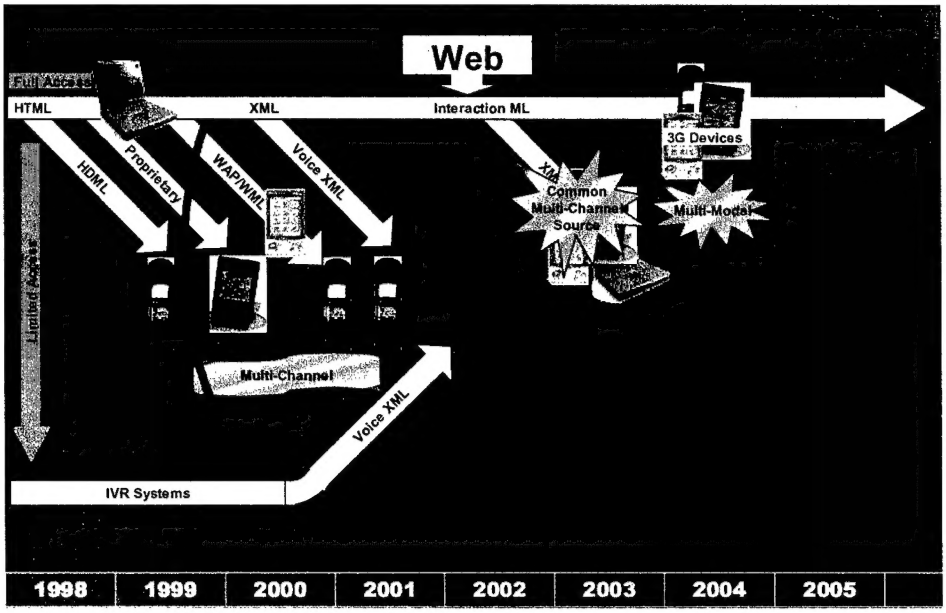


Figure 1 Web Access Roadmap

IBM had complementary ideas of Web site aggregation and a multi-modal dialog markup language technology that together provides a framework that allows a common markup across all devices, with multi-modal access. Though not yet released by W3C, the plan is to begin working on a Multi-modal Dialog Markup Language specification early this year. Presently there is a W3C Voice Browser working group that has a working draft that was developed December 1999. However, this specification addresses only spoken dialog. This committee is now interested in multi-modal dialog. It is envisioned that the new multi-modal working group will establish a common approach. In working with IBM, together we believe this will involve further separation of the Web modal to include an interaction layer.

Overtime, other battlefield information access functionality needs of data objects, fuselets, query, publish, subscribe, analyze and linking will become available. The next natural step for XML is to include information push technology. The battlefield application is not alone with the desire to manually and automatically fuse information, review supporting information and publish it to a subscriber; investment, travel and industrial communities all have the same needs. People want to be informed instead of needing to request information because of time efficiency. Recently, investment and travel alerts capabilities demonstrate this trend, however these are not pure XML solutions, if at all. The W3C work on XForms, Xlink, XPath and SOAP demonstrates that the commercial industries objectives are to this end.

3 Objectives

The underlying objectives laid out in the Phase I proposal was to define an XML framework for the World Wide Web that provides the concept of "write-once, publish-anywhere" for all Internet compliant devices. The objectives were to assess the near-term possibilities of a Multi-Device XML (MDX) framework utilizing XML and an XSL style sheet processor host, and a future Web model that includes an interaction layer.

In the near-term and future scenarios, XML technologies, pervasive browsers, device technologies, and connection bandwidths were considered in determining compatible and acceptable solutions for a common, scalable solution for the Phase II prototype development and demonstration project. The solution must be able to accommodate not only traditional devices, like the desktop computer, but the solution must be scalable, adaptable, and portable to accommodate non-traditional devices like the mobile phone. With voice interaction becoming common across devices, research was conducted to develop concepts for a multi-modal access, giving users the ability to traverse the web with mixed voice and traditional interfaces.

To this end, research focused on concepts to transcode content to device specific markups and a Web model to support mixed interaction. Transcoding content from both legacy and new Web sites to multi-modal devices were studied to allow a migration approach in the hope of market acceptance. Illustrated in Figure 2 is the concept for both the new and legacy Web model: the HTML legacy Web page is downloaded and transcoded into a domain specific Web model; and the new Web page starts at the domain specific Web model. The target

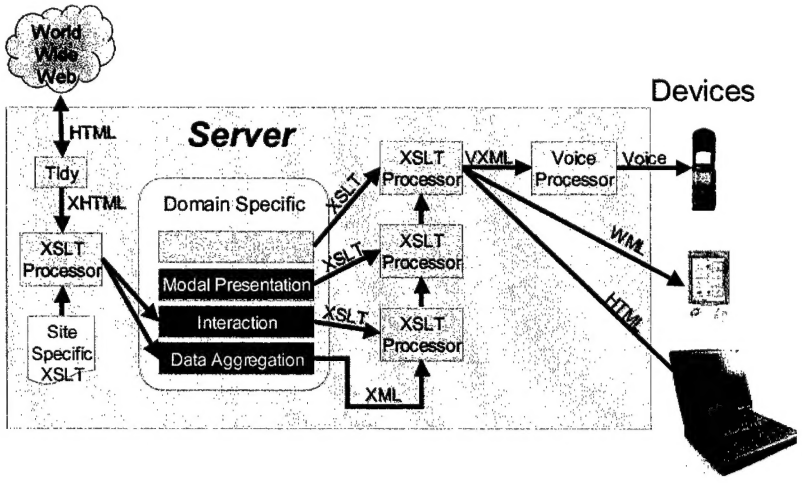


Figure 2 Web Model

Web page is built through a series of XSLT processing steps starting from the aggregated data, adding interaction, transcoding to the target markup and modal presentation, and tailoring with specialized presentation.

To ensure a sound Phase II foundation, risk mitigation was conducted through both research and prototyping. Research involved all applicable XML technologies and concepts for separation of interaction and presentation. Prototyping involved XML key components of VoiceXML and XSLT. The prototype involved filtering Web pages for data and interaction content that is inserted into a Voice XML, XHTML, or WML document. The filter allows new Web content to be created from legacy code.

The most difficult objective to obtain is a truly "write-once publish-anywhere" across devices because of the large mix of browsers in use today. This dream will fall short until a common markup is acceptable by all devices/browsers – a unique transcoding layer will always be required. The objective is to minimize the complexity and number of unique syndications required.

4 Work Performed and Results Obtained

Since the writing of the Phase I SBIR proposal, Industry developments have performed some of the planned work, and as a result helped mature some Phase I associated objectives. The most significant of this work was preformed by the VoiceXML Forum and the World Wide Web Consortium (W3C) XML Architecture and Voice Browser development. The technologies researched and solutions prototyped during Phase I rode the cusp of the industry development of multi-channel authoring. XSLT technology allowed transcoding from XML to anything, providing a path from HTML to today's markup languages comprising HTML, WML and VoiceXML. As demonstrated, this solution improperly suggests a "write-once publish-anywhere" solution as often seen in the industry and is very complex to author from content to presentation. The difference in modal dialog causes interaction to be laced in both of the acclaimed content and presentation layers of XML as illustrated in Figure 3. Because interaction is mixed with presentation, the number of documents per page has the potential to increase to the number of channels, falling far short of the "write-once publish-anywhere" goal. Furthermore, with this model, single channels are published, either HTML to a Mozilla-like browser, WML to a WAP enabled phone or (Personal Digital Assistant) PDA, or VoiceXML to a VoiceXML translator. These channels have been developed by separate groups in the industry with a specific device in mind, resulting single-channel, device-piped solutions.

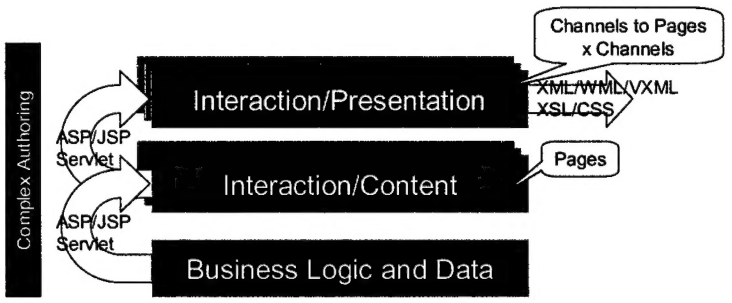


Figure 3 Emerging Web Model

Our study, in collaboration with IBM, developed a model illustrated in Figure 4 that approaches the "write-once publish-anywhere" goal and provides multi-modal. Providing a common interaction layer for all modality allows single authoring for the three complex layers comprising interaction, data aggregation and business logic. Authoring is further reduced by introducing a modal presentation layer. The syndication layer provides specialized presentation. This layer may require multiple documents across channels, but requires lightweight authoring – truly presentation only. Further supporting this future Web model is W3C's recognition of a multi-modal dialog markup language need and anticipation of a new Working Group early this year to develop a specification.

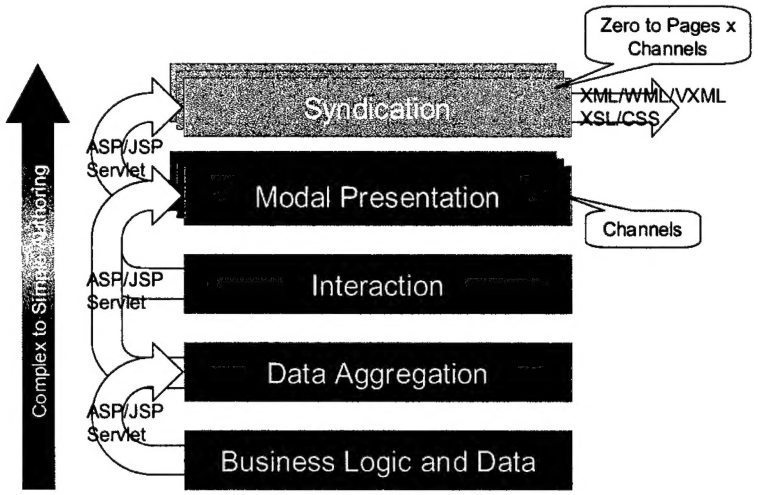


Figure 4 Future Web Model

4.1 Research

The research involved XML and voice technologies and server, tools and browser solutions. Here, technology is used as early developments, not necessarily incorporated into available solutions. Though there is some overlap, there are many areas not yet implemented into these solutions, awaiting standards resolutions and industry acceptance.

4.1.1 XML Technologies

XML technologies researched included interface subjects of data encoding, parsing, database interface, storage/retrieval and transcoding and interaction subjects of XSLT, schemas and forms.

4.1.1.1 Data Encoding Techniques

Converting the data to XML provides several benefits:

1. The message is easy to construct and process using standard XML tools.
2. The contents can be queried using XQuery.
3. It can be transformed using XSLT.

There are potential conflicts with repeated ID attributes that can be overcome by indexing or inclusion using a CDATA section. The benefits of XML are lost if the later is chosen.

For binary data, the commonly used solution is base-64 encoding. This approach provides easy encoding and decoding and the character set of base-64 encoding is valid XML element content. However, base-64 encoding takes up nearly 33% more memory than pure binary representation.

For structured data represented as in the SOAP specification and the XML Protocol, an abstract data model is used. A commonly used abstract data model is the Directed Labeled Graph (DLG). A DLG consists of named nodes and named edges that connect the source with destination nodes. All programming language and database data structures can be expressed as DLGs. Therefore, to handle DLGs in XML three things are required:

1. Given metadata of an abstract data model, a DLG model could be created from where an XML schema is created.
2. Given an instance graph of the data model, XML can be generated that conforms to the schema – serialization.
3. Given XML that conforms to the schema, an instance graph that conforms to the abstract data model schema can be created – de-serialization.

XML Metadata Interchange (XMI) defines this space.

SOAP defines its own set of encoding rules that are very detailed, taking up 50% of the standard. SOAP's encoding model depends heavily on XML schema: ID/IDREF attributes handle multiple references.

4.1.1.2 XML Parsing

Solutions may include more than one technique to benefit from the advantages of each.

There are two major types of XML (or SGML) APIs:

1. Tree-based APIs; and
2. Event-based APIs.

A tree-based API compiles an XML document into a tree structure to allow an application to navigate that tree. The Document Object Model (DOM) working group at the World-Wide Web consortium is developing a standard tree-based API for XML and HTML documents.

An event-based API reports parsing events, such as the start and end of elements, directly to the application through callbacks and does not usually build an internal tree. The application implements handlers to deal with the different events much like handling events in a graphical user interface. The Simple API for XML (SAX) is a standard interface for event-based XML parsing, developed collaboratively by the members of the XML-DEV mailing list currently hosted by OASIS. SAX 2.0 was released on Friday 5 May 2000, and is free for both commercial and non-commercial use.

Both the Sun and IBM libraries provide DOM and SAX parsing strategies.

OmniMark uses a third model: hierarchy-based parsing. Where a SAX parser treats the beginning of an element as one event and the end as a separate event, OmniMark treats the occurrence of an element as a single event that fires a single rule. Since elements can contain nested elements, this leads to the creation of a hierarchy of fired rules.

4.1.1.2.1 Document Object Model

The Document Object Model (DOM) is an application programming interface (API) for HTML and XML documents. DOM exposes data as a tree structure composed of nodes that with the programming interfaces enable applications to traverse the tree and manipulate its nodes. The nodes are defined as a specific node type, which defines valid parent and child nodes for each node type. The most common node types are element, attribute, and text. Attributes are a special part of the model because they are not considered child nodes of a parent. A distinct programming interface, "named node map," is provided for the attributes.

The W3C defines two DOM programming interface groups: fundamental and extended. The fundamental interfaces include those needed to write applications that manipulate documents. The extended interfaces include those that might make it more convenient for developers.

DOM is useful for a wide range of applications, but often put a great strain on system resources if the document is large. Furthermore, some applications need to build their own, different data trees, and it is very

inefficient to build a tree of parse nodes, only to map it onto a new tree. However, for small applications, DOM provides an efficient in-memory tree.

4.1.1.2.2 Simple API for XML

SAX2 is a new version of the Simple API for XML that incorporates support for Namespaces, filter chains, querying and setting features, and properties in the parser.

To understand how an event-based API can work, consider the following sample document:

```
<?xml version="1.0"?>
<doc>
   <para>Hello, world!</para>
</doc>
```

An event-based interface breaks the structure down into a series of linear events:

```
start document
start element: doc
start element: para
characters: Hello, world!
end element: para
end element: doc
end document
```

Application handles these events just as it handles graphical user interface events: there is no need to cache the entire document in memory or secondary storage. It is possible to construct a parse tree using an event-based API, and it is possible to use an event-based API to traverse an in-memory tree.

DOM parsers requires the entire document to be "well formed" or parsable before it can be read. This isn't true for SAX, which parses only a portion of the data at a time. The SAX parser or chaining together smaller DOM trees will be considered when the data set is large.

While DOM was first, and is more formally standardized by the W3C standards body, the SAX model though less rigorously standardized, is supported by many parsing venders.

4.1.1.2.3 OmniMark

OmniMark is a rule-based language. Program execution begins in a "process" rule. Within a rule parsing is initiated with a do xml-parse construct:

```
Process
    do xml-parse instance scan file #arg[1]
        output "<HTML>%c</HTML>"
    done
```

The do xml-parse construct sets up the parser for parsing, and the parser is started when the "%c" occurs in the string. Then what follows are element rules such as:

```
Element "memo"
    Output "<BODY>%n<H1>MEMO</H1>%n%c<BODY>"
```

Here, the "memo" element of the XML document corresponds to the "BODY" element of the HTML document being created. This rule is fired because of the parsing initiated by the "%c" in the do xml-parse statement. A stack of element rules continues to build from here.

XSLT copies the basic processing approach of OmniMark. XSLT's "apply-templates" command is similar structurally to OmniMark's "%c". However, XSLT's implementation tends to use DOM parser, meaning the whole document must be parsed before processing begins.

4.1.1.3 Database to XML Interface Methods

One technique that companies use to simplify development efforts is to map relational database tables onto business objects in an object-oriented language. This approach provides client code that is less dependent on the physical database schema. Tools exist to convert between the physical structure of a database and the logic of a business model.

Another solution is to write code for objects based on the database schema. This is simple if the schema is not very complicated or only a few tables are involved. This becomes problematic if a hundred or more tables are involved. The problem is a logistical one but can be very time consuming to make all the data entries.

A better solution would be to generate the business objects directly from the database using an automated process. Business object source code can be generated using XSLT. The process involves obtaining the database schema using the Java Database Connectivity (JDBC) API. The information then needs to be grouped into an XML tree using the DOM, resulting an XML document that represents the database schema. The database schema represented in an XML document can then be converted to a low-level database object using XSLT. Issues of name conflicts and character compatibilities need to be resolved. XSL's lack of group operations may point to a variant solution of using SAX or DOM to parse the XML input and to generate the code from a program instead of XSLT.

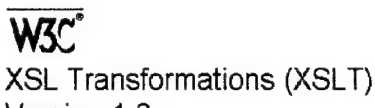
4.1.1.4 Research XML Storage and Retrieval Task

XML data persistence can be accomplished by manipulating and preserving XML, by using object-oriented techniques, Java, and a file system. Whereas the file system is the easy method, it lacks the features of extensive searching, access and versioning. Worst case, if the DTDs and style sheets for a series of XML documents, 90% of the data stored will be duplicated. Today, there are systems based on well-known relational or object-oriented technologies and ones that store native XML.

The object-oriented model provides a tight connection between data and function and there is a high degree of flexibility in subscribing data types and representing relationships. The relational database model is based on a two-dimensional model were data is stored in columns and rows with keys that establish relationships. Objects cannot be stored as entities in RDBMS; this requires a change from an object-oriented schema to a relational schema. When objects are stored, references to the embedded objects must be converted to keys – a programs responsibility. In addition, pointers must be initialized when objects are loaded. The operation necessary to return a child node became overly complicate with deeply structured XML instances. Storing the XML instance as a binary object is not a solution because flexibility is lost. An approach to avoid the problem of storing XML documents directly into RDBMS is to convert the XML document into sets, and to restore the XML instance in application logic.

In the near future, storage systems will be able to store XML in its natural format and manage the DTD and XSL data representation of content. Here, the flexibility nature of XML will be retained while providing optimization needed for storage and retrieval. Technologies exist today, but not broadly applied because the specifications for accessing the data is not finalized or widely adopted. A key example of this is the XML Query Language (XQL) specification that enhances the XML data model and provides a set of query language similar to SQL. Unlike SQL, XQL is limited to operating on a single document or fixed set of documents. However, it can select documents and subset of documents based on conditions defined on document structure and content to create new documents.

4.1.1.5 XSLT



Extensible Stylesheet Language Transformation (XSLT) is the language for transforming XML documents. XSLT was introduced through the beginnings of another W3C recommendation, XSL. XSL is a formatting language that has not been finally approved by the W3C. Included in the original ideas for XSL was the beginning of XSLT. XSL had included with it a pattern language, which would match a pattern with an element within an XML document. Around the same time that XSL was being worked on, a proposal was submitted to the W3C concerning querying, called XML Query Language. The XML Query Language required the use of XSL pattern language. The XSL pattern language was used as the basis for a general query mechanism for XML documents within XML Query Language. In May of 1999, the W3C unified the core semantic models of querying into one recommendation, XSLT. XSLT was approved as a formal recommendation by the W3C on November 26, 1999.

XSLT can transform XML documents into other XML documents, HTML documents, or text documents. XSLT is a language based on XML; therefore, it is an XML language that transforms XML documents. Due to XSLT being an XML language it does not need to be compile to run. XSLT style sheets are run through a processor along with the XML document. The processor performs the requested actions of the style sheets on the XML document.

4.1.1.5.1 XSLT Processor

This approach required us to use an XSLT processor and prior knowledge from work on another contract, we choose IBM's XML Parser for Java. It is an XML Parser completely written in java. The program provides classes not only for parsing, but also for generating, manipulating, and validating XML documents. A processor class can run all of these classes. The processor class can be called separately or written as a piece of a program.

4.1.1.6 Document Type Definition

In order for a parser to understand XML documents, the elements and attributes of the document must first be defined. For years Document Type Definition (DTD) have been used to define the grammar of an XML document. DTDs are created for both single documents and for a class of documents such as an invoice. A DTD defines the elements, their relative parent-child relationships, and their required attributes and data contents. All XML documents are required to meet a well-formedness constraint with respect to the DTD – the document is validated against the DTD. If the document does not match the definition then it is not determined to be well-formed.

In recent years, with XML becoming more and more data oriented, DTDs have started to become outdated because of limitations on what can be defined due to the DTD written format. For instance, DTDs cannot define xml namespaces nor data types. For this reason, the W3C has been looking into XML Schemas to define the grammar for markup languages.

4.1.1.7 XML Schema

Originally, XML was designed to handle basic documents. However, in the past couple of years, XML has been stretched into new areas such as databases, inter-process communication, and object-serialization. These documents require data descriptions and a simplicity that DTDs cannot provide. Therefore, the W3C formed a working group to create a new format for defining markup languages. After much input and combining of ideas, schemas emerged. Schemas are written in XML, which first of all enables them to be treated like other XML documents. Second, the XML base allows schemas to describe data and have

support for data typing. Lastly, schemas allow for elements to be arranged together easily creating a complex object rather than just a group of elements.

4.1.1.8 XForms

In 1993, forms were introduced into HTML and since then have become an important part of the Internet. Over time the existing specifications in HTML for forms are outdated. Thus recently the W3C started working on developing new forms, called XForms. XForms will make a break from the old design of forms by separating the purpose from the presentation. XForms are composed of separate sections. One section describes what a form does while the other describes how the form looks. This new design of forms will allow for more flexible presentation choices. Currently, XForms has a working draft, which means that it is still under review by the W3C and other interested parties.

4.1.2 Voice Technologies

XML has recently been stretched into a new direction to include the world of voice over the Internet. Created are new languages to bring the Internet to someone via voice: VoxML™ and VoiceXML. Also created is a new language to aid the building of voice applications, whether over the Internet or not.

4.1.2.1 VoxML™

Early in 1999, Motorola created an XML language to provide an easier way to produce voice applications, called VoxML™. VoxML™ was designed to be like HTML, a common and widely supported platform, for voice applications. The design of VoxML™ allows the voice applications to be in the form of dialogues. The design also allowed them to be similar to the basic building blocks of HTML. Instead of the user reading the text, the text is read to the user via text to speech engines. Navigational controls are much the same as in a browser. For input, the user speaks instead of typing. Alas, VoxML™ never became widely used. However, the concepts used to create VoxML™ went into creating VoiceXML.

4.1.2.2 VoiceXML

Voice XML is a subset of XML used for the creation of voice interaction with no visual capabilities. Coming from a very diverse background, Voice XML combines together the talents of AT&T, Lucent, and Motorola. AT&T and Bell-Labs, before it became Lucent, were working on a phone markup language geared toward touch-tone capabilities. This markup was called Phoneweb. When AT&T and Bell-Labs spilt, AT&T continued to work on Phoneweb, while Lucent, formerly Bell-Labs, started to work on a project similar that focused on Natural Language Applications. Lucent called their language Teleportal. At about the same time, independent of AT&T and Lucent, Motorola was working on a project, called VoxML™, which combined XML and speech recognition. In March of 1999, AT&T, Lucent, Motorola, and IBM combined efforts creating the Voice XML Forum.

From the combining of AT&T, Lucent, and Motorola Technologies, Voice XML was created. Voice XML combines the ability to give touch-tone usage and the ability to process and understand natural language with XML. These qualities produce a strong language with multiple possibilities. Voice XML has the possibility to be used in collaboration with other types of XML on the Internet, thus providing information in a visual and auditory form on a page. Voice XML also has the ability to create voice-enabled applications outside of the applicability of the Internet. Since Voice XML was released as a W3C publication on May 5, 2000, many corporations, large and small, have been implementing Voice XML into their applications.

4.1.2.3 CallXML

A company called Voxeo recently created an XML language called CallXML. CallXML is not designed to process natural language, therefore, it is much more simplistic than VoiceXML. CallXML is designed to be used to describe to a CallXML browser the user interface of a telephone, voice over IP, or multi-media call application. The design of CallXML is supposed to make it easier for web developers to create applications that can interact with and control any number or types of calls. The call applications can include:

- The initiation and routing of a phone call via telephone or voice over IP
- Interaction with and response to touch-tone based entry and selection via telephone or voice over IP
- Call applications that included support for additional media, such as faxes and video

CallXML has not been submitted to the W3C as a proposal.

4.1.2.4 Dialog

Dialog is not a new XML language, but instead a list of requirements for spoken dialog interaction. There are three main types of requirements addressed for the voice markup language: modality, functional, and format. The type of modalities supported by the markup language for system output and user input is the interest of the modality requirements. For the functional requirements, the behavior resulting from a voice markup interpretation is the concern. Lastly, the format requirements set the syntax of the voice markup language. The ability of the voice browser to interpret the voice markup will affect the requirements. The type of environment, desktop versus telephony-based, will also affect the requirements.

4.1.2.5 Multi-modal Dialog

The W3C Voice Browser working group is creating requirements for Voice Browsers to follow. The requirements are broken into sub-sections, of which the Multi-modal is one. The Voice Browser working group is considering adding multi-modal capabilities to voice browsers. The browser would have to be capable of handling one or more of the following speech modes: speech recognition, speech synthesis, or prerecorded speech. The browser would also have to handle one or more of the following input/output modes: DTMF, keyboard, small screen, pointing device, or some other type of input/output. The document addresses applications in which both speech input and speech output is available. The other main item the document addresses is the ability of a markup language to use spoken dialog interaction along with other modalities.

4.1.3 Interaction Separation Concept

The way we interact with the Web today is very uncommon with how humans interact with each other. When humans speak, each can initiate the course and change the topic of the dialog – personalities aside. A traveler can let an agent know that she would like to go to New York from February 15th to the 19th. The agent then could ask from what city. The traveler responds with Baltimore as the departing city with a follow up request of whether the Ravens are still winning the Super Bowl. Here, both the traveler and agent are initiating dialog, and the traveler can mix in other topic requests allowing a richer and more efficient exchange of information.

With today's browser technologies dialog is very much controlled by the machine. With a keyboard/mouse interface the traveler may be asked to fill out a form in completeness with the form reappearing when submitted if not complete. With a limited display or voice-only device the traveler will be taken through a series of questions from "what is your destination", "where will you be departing from", to " the total cost is $456.76, would you like to purchase the ticket". In either case the computer asks for the information requested and the traveler has no means of entering additional information, let alone a mixed topic request.

Alike, obtaining content using today's browser technologies is not how humans obtain information from one another. A requester of information may need to answer some follow up questions in providing clarification, but the process is straightforward. With a browser we either wander through hyperspace or are provided a limited content. With a keyboard/mouse interface you click and wait through hyperspace traversing pages and pages before reaching the desired information. On the other end, when using a limited keypad/voice interface much of the Web content is not reachable.

Today, Web interaction is browser-centric where users are forced to interact in a non-conventional manner with the Web because browsers are very sequential; they do not provide content persistency. Instead they provide limited Web site scope via linked pages. A user-centric approach must be taken to support the concept that humans desire to interact with machines as if they were humans as illustrated in Figure 5. This will involve understanding how humans interact with information across modalities.

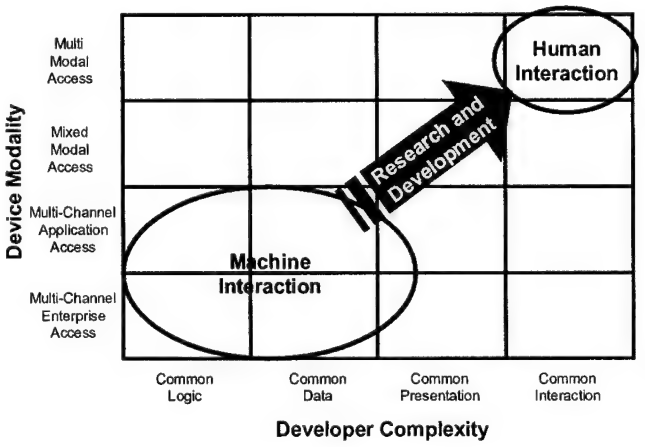


Figure 5 Machine to Human Interaction Migration

However, interaction across modalities can be daunting given the combination of modalities used, and the complexity of how humans prefer to interact with compound information and mixed-initiatives. The concept is to separate the interaction into its own layer where it can be managed separate from data and presentation as illustrated in Figure 5. This is analogous to a dialog flow diagram of an entire Web site were each state represents a page of dialog. Interaction of the entire Web site in a single document allows client-side multi-modality synchronization, for example in a voice browser companion for the desktop as described in the next section. Here, multi-pages can be traversed with a single voice command instead of sequentially downloading subsequent pages. The W3C activities of XForms and Multi-modal Dialog, and IBM's research are showing promise of industry recognition in this area.

4.1.4 Voice Browser Companion Concept

Voice browsers were originally created to assist the visually impaired to traverse Web content, comprising of basic screen readers such as Simply Talker and the Hal Screen Reader. The Voice Browser Companion goes beyond this; it provides an integrated solution of the traditional keyboard/mouse browser with a voice browser. The idea is to allow users to transcend down to specific Web content in a single request beyond what is displayed on the screen. The best example of this is to obtain the support phone number of a particular Web site. This often takes five to twenty clicks and waits in guessing where to go. With a single spoken command the wrong branches and nested pages are skipped and the page of interest is displayed.

The integration of the Voice Browser Companion with traditional browsers can take on three forms:

1. Browser independent, client side
2. Browser independent, server side
3. Browser dependent

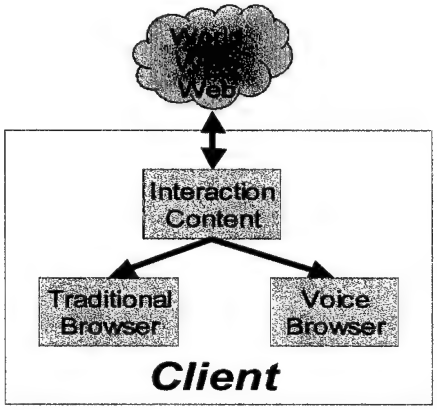


Figure 6 Browser Independent, Client Side Voice Browser Companion

The Browser independent, client side approach integrates existing browsers, unmodified, into the Voice Browser Companion framework. This allows adoption into a pervasive infrastructure of IE, Netscape and WML browsers. This involves a tiered layer where the top layer task is to provide interaction content and to synchronize the browsers as illustrated in Figure 6. The Voice Browser interacts with the Web content through the DOM interface with no modification to the traditional browser. For Web content not designed with site interaction, minimal voice browsing capability is provided through link recognition of the current page. The ability to traverse multiple pages can only be achieved if the interaction content process has knowledge of the complete Web site map. The interaction content process provides default mapping of traditional links to default voice browser methods in the absence of interaction content. This approach has the advantage of requiring no server side modification and interfacing with traditional browsers unmodified. Hence, this capability can be used across the entire World Wide Web. The disadvantage is the increased memory and CPU required on the client that is an issue for pre 3rd generation devices.

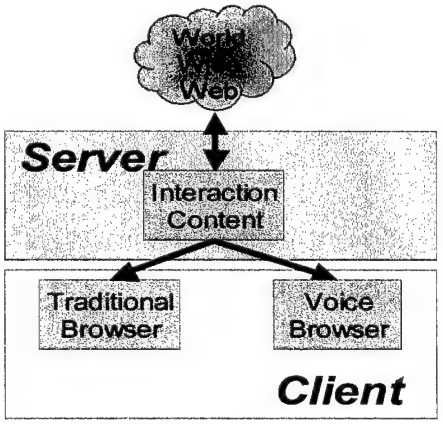


Figure 7 Browser Independent, Server Side Voice Browser Companion

The second approach pushes the interaction content process to the server side as illustrated in Figure 7. This requires a proxy site to provide voice browser content of legacy sites, which may be viewed as a disadvantage. Synchronization becomes a problem because of latencies while downloading Web content.

The best, seamless approach is to integrate the voice browser companion into the traditional browser. Candidates would include IE, Netscape, AOL and the popular WML browsers. The same concept can be used to map default Web content to default voice interactions along with accepting new interaction Web content sites. The disadvantage is that voice browsing depends upon the browser used – traditional browsers lacking interaction markup engine will not recognize new interaction Web content sites.

4.1.5 Server Solutions

The idea of a speech server has only recently become hardware that is highly desired. Before the grandiose ideas of creating complete voice systems, speech recognition was only for specialty systems. These systems did not require the power of a complete speech server. Due to the required power and speech system knowledge, there are only a small number of companies who have developed speech servers. The requirements for the speech server are:

- a natural speech recognition engine
- a text-to-speech engine
- the ability to interpret Voice XML
- the ability to peruse the Internet

There are primarily two solutions for speech server, IBM's WebSphere and Lucent's Speech Server. Nuance also makes a speech server, but is more widely know for their speech recognition engine and speech development tools.

4.1.5.1 WebSphere

The approach taken to provide universal access from a legacy site required a program to run servlets. An application server runs servlets along with providing many other useful tools. The edition of IBM's WebSphere Application Server used allowed easy access to a database and supported eXtensible Stylesheet Language (XSL). It also allowed the use of JavaServer™ Pages. Advanced editions of WebSphere provide Enterprise Java Beans (EJBs) and CORBA support. WebSphere runs on an HTTP server based on Apache Web. The HTTP server can run without WebSphere, which only provides basic web hosting options.

Included in the WebSphere family line, is a voice server. IBM's WebSphere Voice Server is designed for a telephony environment, using Voice Over IP (VOIP) as their protocol. IBM has had a diversity of experience creating text readers and speech recognition systems. IBM's Voice Server provides a text-to-speech engine, a voice recognition engine, and a Voice XML browser. The voice browser can run in correlation with the application server or it can run separately. The Voice Server uses a combination of Voice XML and mixed-initiative dialog to create the environment for the user. This allows the user to speak naturally as if truly in a conversation.

4.1.5.2 Lucent Speech Server

Lucent Technologies has had an abundance of experience creating text readers and speech recognition systems. The Speech Server they have designed is a natural speech recognition system. It is designed for a large-scale network requiring a high-density solution in a telephony environment. Lucent's Speech Server offers Voice-Activated Dialing, Auto Attendant, Directory Assistance, and Intelligent Personal Agent as services. The speech server was designed with the latest technology in mind by using a new markup language, Voice XML. Basic telephony capabilities like dual tone multi-frequency (DTMF) are also provided with the Speech Server. This service can be incorporated into Voice XML documents, thus making the document more flexible by offering both voice and DTMF to the user as input capabilities.

4.1.5.3 NUANCE

Nuance is a company that works with natural voice interface software for telephone networks. Adding to their list of voice interface software, Nuance has developed a Voice Web Server designed to run voice sites and services accessible by a telephone. The core of the Nuance Voice Web Server is Nuance 7.0, a speech recognition server. Also incorporated into the Nuance Voice Web Server is an interpreter for Voice XML, built on the 1.0 standard for Voice XML. Nuance Voice Web Server allows for applications like, voice activated dialing and telephone access to intranet or web portals, to be created. Not incorporated in the Nuance Voice Web Server, but to work in correlation with Nuance developed a voice browser, called Nuance Voyager. Nuance Voyager runs on the Nuance Voice Web Server. Special features like, Hotword and Active Help, make navigation between sites easy with Nuance Voyager.

4.1.6 Tools Solutions

There are not many speech development tools available; Lucent and IBM have some tools included in with their speech servers. However, most notably Nuance created several speech development tools that are separate from their speech server. Not only are there tools for voice applications, but there are also tools to help in the transition between legacy HTML code and XML code. Each of these tools can have a great impact when working with HTML and Voice technologies.

4.1.6.1 NUANCE Tools

Nuance offers several tools to aid in design, prototyping, development, customization, testing, and tuning of speech recognition and voice authentication applications.

4.1.6.1.1 Nuance V-Builder

Nuance V-Builder is a Voice XML development tool. Using two new standards, Voice XML and Nuance Speech Objects, which are reusable speech components, Nuance V-Builder creates an easy to use drag and drop environment.

4.1.6.1.2 Nuance Foundation Speech Objects

Nuance Foundation Speech Objects are speech components that support speech recognition and voice authentication. The speech objects are free source and are extremely portable since they are designed to be implemented as JavaBeans.

4.1.6.1.3 Nuance Grammar Builder

Nuance Grammar Builder is a graphical tool intended to allow developers an easy way to create and test grammar from a PC.

4.1.6.1.4 Nuance V-Optimizer

Nuance V-Optimizer is a tool used to understand customer response to voice interface applications. Reports and charts are created to help developers improve voice interfaces.

4.1.6.2 Tidy

One of the many contributors to the W3C decided to create a utility to aid in editing HTML code. Tidy is designed to fix mistakes made while writing code. Tidy is able to fix a lot of minor problems that can be easily over looked. For example, Tidy is able to clean up missing or mismatched end tags. In an attempt to help the designer learn from past mistakes, Tidy will also create and error log pointing out items that need to be worked on.

4.1.7 Browser Solutions

The modern web is composed of extravagant Flash presentations, lively animations, ornate movies, and nifty images that are awkwardly tied together with HTML. Tomorrow's web promises to be full of sophisticated presentation backed by a semantic web using eXtensible Markup Language (XML) and other emerging standards from the World Wide Web Consortium (W3C). Between now and then, we are forced to use today's web browsers that have yet to fully comply with the standards laid out by the W3C. Here is a list of the standards/abilities used to describe the browsers below.

- HTML – HyperText Markup Language 4 (1998).
- CSS - Cascading Style Sheets (Level 1 – 1996, Level 2 – 1998).
- DOM – Document Object Model (1998).
- XML - eXtensible Markup Language (1998)
- XHTML - Extensible HyperText Markup Language, a reformulation of HTML in XML (2000).
- XSLT - Extensible Stylesheet Language Transformations (1999).
- Java – The inclusion of applets in web pages (not a W3C standard).
- Script support – The scripting language supported by the client (JavaScript is standardized by the European Computer Manufacturers Association, ECMA)

4.1.7.1 Traditional Browsers

The graphical web browser has been around for several years and in that time only a few have stood the test of time. Netscape was the first browser to stand out and shortly there after came Internet Explorer. Opera is the most recent to be added to the list of strong standing browsers. Other browsers have been created, but they did not make a lasting mark in the market and quickly faded away. Mozilla is an emerging browser that will be replacing Netscape.

4.1.7.1.1 Netscape

The browser that originally grew along with the development of the Internet has now become outdated and is experiencing a decline in the market share. It only supports HTML, JavaScript, Java, and very basic CSS. Web developers are often forced to write around its incompatibilities and develop less robust sites. Currently, development of the 4.x code base is limited to security fixes and there are no plans to support emerging web technologies. With the current expansion of XML technologies, it is expected that Netscape 4.x will slowly disappear over the next few years. The worst projections suggest that the shortcomings of 4.x will haunt web developers for the next three years.

4.1.7.1.2 Microsoft Internet Explorer (IE)

Easily the most popular web browser, IE has gained its fame from being distributed with Windows (98/NT/2000/ME) and AOL. The most recent release (5.5) includes support for the latest web technologies including XML, XSLT, XSL, CSS, and DOM. The only disadvantage is that it is not 100% compliant to any of those standards. Another issue with IE's standards compliance is varying support in the same version on different platforms. Additionally, Microsoft has a poor record for providing security fixes for their browser. Comparatively, IE 5.5 has the best support for CSS and HTML. However, when designing a standards compliant web site and only using IE, it is very easy to use proprietary MS extensions to compensate for IE's inadequacies. For example, the XML tag is not part of any HTML specification and is not supported by any other browser than IE 5.0 and later. The XML tag allows IE to break out of HTML jumping straight into XML and then back to HTML. This tag allows IE to have a capability that the other browsers do not.

4.1.7.1.3 Opera

A latecomer to the browser war, Opera Software in Norway created a browser with the vision of running it on older machines as well as new and increasing usability for the disabled. Although it is far from the most popular browser, Opera does have a small but very loyal customer base. It fully supports some of the latest web technologies, including HTML, JavaScript, Java, XML, CSS, DOM, and even WML (Wireless Markup Language). The biggest limitation is that most web sites are developed to be viewed with one of the big two browsers (IE or Netscape) therefore Opera does not support the proprietary features found in either browser. This results in poor presentation of complex web pages that is often viewed as a fault in the web browser.

4.1.7.1.4 Mozilla

Mozilla is the original nickname for the source code of Netscape and has now become the name of the project that grew from the open release of Netscape's source. In the course of developing the next release of Netscape, the Mozilla project completely tossed the old source and began writing the "next generation" browser. Throughout the past two years, Mozilla.Org has released target milestones (one every few months) and nightly builds. The latest milestone was M18, which had support for the latest web standards and binaries for nine platforms (Win32, MacOS, Solaris, Linux, OpenVMS, FreeBSD, HP-UX, OS/2, & Tru64 Unix) with continuing work on BeOS, Irix and a few others. Mozilla is also being translated to over 40 different languages. Mozilla currently implements full support for HTML, CSS, DOM, XML, XHTML, JavaScript, and the APIs for Sun's Java plug-in. Additionally, Mozilla utilizes a new interface design, XML-based User Interface Language (XUL - pronounced "zuul"). XUL is the foundation that provides interface customizations and interchangeable skins. Furthermore, outside contributors have aided in the evolution of the project and have created extensions for XSLT, Scalable Vector Graphics (SVG), and MathML, but those extensions have yet to be added to the regular builds. Current estimates suggest a release of Mozilla 1.0 (including extensions) in the mid 2001 timeframe.

4.1.7.2 Mobile Browsers

Recent developments in small devices and cellular communication have set the stage for a new suite of browsers. A standard option on most cellular phones is the ability to connect to the Internet. Almost all of these phones use their own proprietary browser that is compatible with WML (Wireless Markup Language), cHTML (Compact HTML), or some other proprietary markup language. These markup languages used for mobile computing are not standardized by the W3C; they have been created by a group of industry

representatives called the WapForum. Handheld devices also offer the ability to connect to web clippings or to the Internet via one of the following browsers:

- Kbrowser - A browser for multiple mobile devices that supports WML.
- WAPman - A browser for both Windows and Palm OS that supports WML.
- WinWAP - A browser for windows that supports WML.
- Klondike - A WML browser that runs on Windows and Windows CE.
- KWML - A WML browser that runs in Sun's KVM (a micro edition Java virtual machine).
- Pocket IE - A HTML for Windows CE.
- Up.Browser - The WML browser found in numerous cellular phones.

4.1.7.3 Voice Browsers

Alternative web interfaces were originally created to assist the visually impaired. However, recent advancements in voice technology have lead to another group of browsers that are voice enabled. These new browsers are the evolutionary step from the screen readers of the past such as Simply Talker and the Hal Screen Reader. For example, IBM's new WebSphere Voice Server runs in Windows and provides the means of interacting with VoiceXML web pages. And Lucent Technologies' VoiceXML Development System links a phone number with a VoiceXML page on the web using Lucent's own text-to-speech and voice recognition technologies. These innovations as well as others imply endless possibilities with multi-modal interaction to the future Internet.

4.2 Prototype

4.2.1 Voice Proxy Server Architecture

During Phase I, we researched universal methods to access the World Wide Web with the goal of "write-once publish-anywhere". This involved identification of syndication and Web content standards and technology development to close the gap between the goals setout and the pervasive device capabilities found in the marketplace. HTML has been the language most used on the World Wide Web, however in the past few years other languages have been introduced that provide access to Web content. Most notably are the wireless cell phones and PDAs, as these became more popular, a demand for Web access increased. The new languages developed to attach these devices to the World Wide Web are in some cases similar to HTML and others not. Further compounding the problem is the fact that there are several languages trying to achieve the same goal, through different means. Many of these languages became proprietary and not widely used once created. Results of the investigation showed both WML and VXML to be the most promising languages to continue into the future based on industries adoption of these standards. Hence, for Phase I, we chose WML and VXML as the languages to represent for these smaller devices, within a demonstration.

A server-side process was developed to transcode legacy Web content to these devices languages. However, as devices become more powerful, some of the processing logically will move to client-side to support multi-modal interaction. A universal access servlet portrayed in Figure 8 was developed to transform legacy HTML code into XHTML, WML, or VXML. This was implemented in Java with technologies from IBM, Apache and W3C. Amongst all the markup languages, HTML is one of the few that is not based on XML, for this reason the W3C goal is to phase out HTML by replacing it with XHTML. This is essentially a cleaned up version of HTML that is XML compliant.

As illustrated in Figure 8, the servlet is called from the URL bar, entry field or Voice Server depending upon whether a desktop browser, PDA, or cell phone is used. An augmented URL was developed to point the request to this universal access servlet. First, the URL is passed through a function that parses the string and matches all the parameters of the URL string with their values. The modified URL string is utilized to retrieve content from the World Wide Web. The retrieved data is then stored to a file for processing.

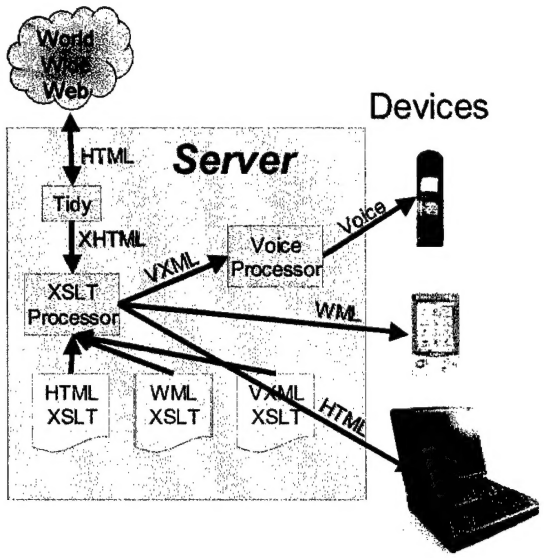


Figure 8 Universal Access Servlet Prototype

The source file is then sent through a program, called Tidy, which cleans up the HTML and creates XHTML. Tidy is a free program created by the W3C. Tidy adds end tags to elements, even empty elements, that do not have end tags, changes the case of upper case elements and attributes to lower case, and will remove old tags or update old tags to the newest specification. Even without the XML declaration and namespace, the cleaned up HTML is still valid as an HTML file. Tidy does a great job of cleaning up the HTML, however if the HTML is so poorly written it will over compensate and produce undesirable results.

The servlet detects the browser type making the request. This is important to transcode the source code to the language that is compatible with the browser. However, determining the client browser type is not always possible. Furthermore, there are so many browsers that it is difficult to handle their varying intricacies and to adapt to their unique capabilities. These browsers are introduced into the market almost daily. The browser types are placed in with the HTTP request in a parameter called 'user-agent'. The acceptable types of files (HTML, VXML, and WML) are determined by the 'user-agent' parameter. This determination is derived from the name of the browser by programmatically selecting the type of data acceptable based on knowledge of what that browser can accept. If the browser type is not known to handle HTML or VXML, then WML is assumed. Unfortunately, the 'accept' parameter cannot be used because it does not always specify all the mime types it accepts, which would leave the servlet to guess as to which file type to send. Many times the value will only be '*/*', which specifies anything, but that is not true because they do not all accept WML files. Once the browser type is decided the modified file is then passed through an XSLT processor. A style sheet is written for a conversion to WML, VXML, and XHTML. The XSLT processor writes the modified information to a new file, which is then sent back to the browser.

The ETrade site was chosen to demonstrate the abilities of the servlet. Therefore, each XSLT style sheet was written with something specific in mind. The XHTML/HTML style sheet only needed to add the site information at the front of any <a> element reference that started with '/cgi-bin'. The WML style sheet parses only the relevant information. The ETrade index page has a lot of information, and because of the limited size many WML browsers can handle only the most important could be passed on to a WML browser. Therefore, the WML style sheet parsed the index page for the main links like the links to home, trading, quotes & research, account services, portfolios, and banking. For other pages beyond the index page, the WML style sheet would only give you the relevant information for that page. Voice browsers do not have a limit on the size that they can handle, however, to limit the read time, the VXML style sheet was designed similarly to the WML style sheet.

5 Conclusions

Throughout the course of the Phase I project many emerging technologies were researched and prototyped in search of components of a World Wide Web universal access framework allowing multi-device and multi-modality interaction while striving for a "write-once publish-anywhere " objective. Early-on, pliable commercial technologies where sought and industry collaboration with IBM was established. Initially the research rode the cusp of commercial industries adaptation of XSLT to transcode Web content to various browser markups. Though showing promise of automated transcoding, this method falls far short of the "write-once publish-anywhere" objective, and does not address multi-modality aspects.

The research continued in coordination with IBM in establishing a common interaction layer allowing coordinated multi-modality and simplifying presentation content authoring. This focus is extended beyond industries implementation as seen with XSLT because development is in the very early stages – W3C has not yet formed a working group to address multi-modal dialog but has recognized the need. The intent is to help form this solution throughout the course of Phase II through cooperation of IBM and other industry leaders.

Acronym List

API	Application Programming Interface
AUSWC	Advanced Undersea Warfare Concepts
cHTML	Compact HTML
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheets
DARPA	Defense Advanced Research Projects Agency
DLG	Directed Labeled Graph
DOM	Document Object Model
DTD	Document Type Definition
DTMF	Dual Tone Multi-Frequency
EJB	Enterprise JavaBeans
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JBI	Joint Battlespace Infosphere
JDBC	Java Database Connectivity
PDA	Personal Digital Assistant
RDBMC	Remote Database Management Controller
RDBMS	Remote Database Management System
SAB	Scientific Advisory Board
SAX	Simple API for XML
SGML	Standard Generalized Markup Language
SOAP	Simple Object Access Protocol
SQL	Server Query Language
URL	Universal Resource Locator
VOIP	Voice over IP
VXML	Voice Extensible Markup Language
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WML	Wireless Markup Language
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language

XQL	XML Query Language
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformation